RETI DI CALCOLATORI – prova scritta dell' 11/9/2018

Per essere ammessi alla prova orale è necessario ottenere una valutazione sufficiente della prima parte e una votazione totale di 15 o superiore.

Prima parte (15 punti)

Matricola_____Cognome_____Nome_____

Q1. Consideriamo un sistema autonomo che contiene tre server aventi i seguenti nomi simbolici, server web: www.pippo.it (canonico) e web.pippo.it (alias), server ftp: ftp.pippo.it (canonico) e files.pippo.it (alias), e email server: mailserver.pippo.it (canonico) e email.pippo.it (alias), con indirizzi IP 190.168.1.251, 190.168.1.252, 190.168.1.253, rispettivamente. Indicare il contenuto dei resource record relativi ai server su indicati presenti nel server DNS locale dns.pippo.it (che ha indirizzo IP 190.168.1.255).

RISPOSTA: per il server web: **(*www.pippo.it*,A,IN,TTL,190.168.1.251)** e **(web.pippo.it,CNAME,IN,TTL,www.pippo.it)**
per il server ftp: **(** ftp.**pippo.it**,**A,IN,TTL,190.168.1.252)** e **(files.pippo.it,CNAME,IN,TTL,ftp.pippo.it)**
per per l'email server: **(mailserver.pippo.it,A,IN,TTL,190.168.1.251)** e **(email.pippo.it,MX,IN,TTL,www.pippo.it)**

Q2. Supponiamo che al tempo t0 il TCP di un host A abbia già stabilito una connessione con un suo pari B, che abbia 1 segmento "in volo" contenente 1 MSS di dati, che il valore di sendBase sia X e che abbia 1 MSS di nuovi dati da spedire. Al tempo t1>t0, A riceve un segmento S1 che non contiene dati e arriva ad A non corrotto. Al tempo t2>t1 scatta il timeout e A invia un segmento S2. Nell'intervallo [t0,t2), A riceve solo il segmento S1 e non scatta nessun timeout. Indicare i possibili valori del campo AckNum di S1, del campo SeqNum di S2 e la quantità di dati contenuti in S2 nel caso in cui il valore di RcvWin in S1 sia 0.

RISPOSTA*:*
Se AckNum di S1= **X+1 MSS**, allora SeqNum di S2= **X+1 MSS** e la quantità di dati contenuti in S2= **1 byte (zero window probing)** ;
Se AckNum di S1= **X**, allora SeqNum di S2= **X** e la quantità di dati contenuti in S2= **1MSS (non può essere il terzo ack duplicato perché A non invia niente subito dopo t1)** ;

Q3. Due router, A e B, che possono ricevere e trasmettere contemporaneamente, sono posti ad una distanza di 25 Km e sono direttamente collegati tra loro con un canale full duplex. Al tempo T, A inizia a inviare frames lunghi 48 byte a B, e B risponde con ACK di 3 byte, e non ha nessun dato da inviare ad A. A e B trasmettono ad una frequenza di 12,5 Mbps, e la velocità di propagazione è di $2x10^8$ m/sec. Supponendo che B riceva tutti i frames corretti, che tutti i riscontri di B siano ricevuti da A corretti, e che non scada mai il timeout, dopo quanto tempo A *finisce* di trasmettere la *seconda* finestra se a livello LLC, A utilizza stop and wait (considerando la finestra ampia 1 frame), oppure go back N (con finestra ampia 4 frames), oppure selective repeat (con finestra ampia 5 frames)? Per semplicità si considerino i tempi di accodamento ed elaborazione trascurabili.

RISPOSTA: Se A utilizza stop and wait, finisce di trasmettere la seconda finestra dopo **313,36** microsecondi, se A utilizza go back N, finisce di trasmettere la seconda finestra dopo **405,52** microsecondi, se A utilizza selective repeat, finisce di trasmettere la seconda finestra dopo **436,24** microsecondi. **(ritardo di propagazione=125 microsecondi, tempo di trasmissione di 1 frame=30,72, tempo di trasmissione di 1 ack=1,92 microsecondi)**

Q4. Una rete è formata da 7 sistemi autonomi, denominati S_1, S_2, S_3, S_4, S_5, S_6, S_7, che contengono rispettivamente 18, 22, 25, 26, 55, 10, e 39 reti locali e 5, 10, 4, 6, 12, 3, e 9 router. In particolare, 2 dei 6 router di S_4 , $G_{4,1}$ e $G_{4,2}$ sono router gateway, collegati rispettivamente ad un router gateway di S_2, e ad un router gateway di S_7. Qual'è il numero *minimo* di righe di cui è composta la tabella di inoltro *definitiva* di $G_{4,2}$?

RISPOSTA: la tabella di inoltro definitiva di $G_{4,2}$ è composta da **27** righe minimo **una per ciascuna rete locale + 1 se i cammini per tutti gli altri AS passano per lui**

Q5. Una rete *ethernet standard* cablata è lunga 4,5 Km. Se la velocità di propagazione dei segnali nella rete è di $2x10^8$ m/sec, e i frames sono composti da un numero intero di byte, qual'è la *lunghezza minima* dei frames (espressa in byte) secondo lo standard?

RISPOSTA: la lunghezza minima dei frames (espressa in byte) è **57 (per CSMA/CD: deve essere non minore del tempo per trasmettere in 2 rit. propag. a 10 Mbps) oppure 64 (secondo lo standard)**

Seconda parte

E1 (7 punti). Consideriamo un'applicazione che utilizza UDP per realizzare un trasferimento (non affidabile) unidirezionale di dati in cui il mittente utilizza numeri di sequenza per numerare ogni gruppo di dati che invia con UDP. Questi gruppi sono tutti della stessa dimensione. Il ricevente mostra all'utente i dati ricevuti rispettando l'ordine con cui sono stati spediti dal suo pari e tollerando la perdita di al più due gruppi consecutivi di dati. Quando il ricevente riceve l'i-esimo blocco di dati risponde al suo pari

- con un messaggio *ACK(i)* se può mostrare i dati ricevuti all'utente, altrimenti
- con un messaggio *NEED(x)* indicando così che l'ultimo blocco che è riuscito a mostrare all'utente è l' x-1 esimo.

L'applicazione, lato mittente, è costituita da due threads: *interfutente*, che si interfaccia con l'utente umano e inizializza le variabili usate dai due threads, e *interfUDP*, che si interfaccia con UDP.

Descrivere mediante pseudocodice, il comportamento di interfutente e interfUDP, assumendo che utilizzino una finestra di dimensione W (in numero di gruppi di dati). Descrivere il contenuto o la funzionalità delle altre variabili e funzioni/procedure utilizzate. Per semplicità, si ignorino i problemi di concorrenza tra i due threads.

***SOLUZIONE**:*

interfutente

```
base=0;          // base indica il gruppo di dati più vecchio ancora in volo
next=0           // next  indica il prossimo numero di sequenza utilizzabile
while true
{  reqSend(data) ;  // reqSend è l'operazione con cui l'utente chiede di inviare nuovi dati
   if  (next>=base+W)
     {  RefuseRequest()  } //  refuseRequest  è la risposta all'utente se non c'è spazio nel buffer
   else
     {  B[next mod W]=numberData(next,data);  // B è il vettore usato per memorizzare i dati spediti e in attesa di
                                              // riscontro ;numberData(next,data) pone il valore next all'inizio del gruppo di
                                              //dati da spedire (lo numera)
        UDPsend(B[next mod W]);  // UDPsend passa i dati ad UDP per farli inviare
        next++;
      }
}
```

interfUDP

```
forall i acked[i]= false;  //per sapere quali ack sono stati ricevuti (UDP/IP non garantiscono l'ordine di consegna)
while true
{  r=UDPrcv();  // UDPrcv  per ricevere i dati da UDP
   if  ( correct(r)&&(r=ACK(i))  // correct(r) funzione booleana che calcola la correttezza di r
     {  acked[i]=true;
        while ((acked[base]) || (acked[base+1])|| (acked[base+2]))  //per la tolleranza di max 2 gruppi consecutivi
            { base++; }
      }
    if  ( correct(r)&&(r=NEED(x))
     {  if (x>=base) {base=x;}
        for (j=base;j<next;j++)
           {  UDPsend(B[j]);
            }
      }
}
```

E2 (8 punti). Un sistema autonomo S è connesso con internet tramite un unico router NAT R (che utilizza indirizzi IP e numeri di porta). In particolare, R è dotato di due interfacce di rete, II ed IE: la prima è collegata con le entità interne ad S (e che utilizzano il blocco di indirizzi privati 192.168.0.0/18), mentre l'interfaccia di rete IE, che ha l'indirizzo pubblico 102.103.104.105, è collegata con internet. Descrivere, mediante pseudocodice, il funzionamento di R quando riceve un datagram proveniente da S (cioè ricevuto su II). Notare che possono esistere più sessioni esterne contemporanee per uno stesso host di S. Si hanno a disposizione le seguenti procedure:

receive(interf,dg) per ricevere il datagramma dg dalla interfaccia interf;

send(interf,dg) per inviare il datagramma dg sulla interfaccia interf;

lookup(a,b,c,d,e) che restituisce l'indice della tabella *a* in cui si trovano gli elementi *b* e *c*, e restituisce *-1* se nessuna riga di a contiene b, c, d ed e;

changeport(dg,x) che scrive x nella porta sorgente dell'header di livello trasporto di dg;

portalibera() vettore booleano il cui elemento x vale true se e solo se la porta x non è stata usata da R;

trovaportalibera() restituisce un numero di porta attualmente non utilizzata da R.

Per semplicità, si supponga che i numeri di porta siano replicati nel campo options del datagramma: *dg.options.sourceport* per la porta sorgente, e *dg.options.destport* per la porta destinazione, che la tabella di traduzione *NATTAB* sia sufficientemente grande da permettere sempre aggiunte di nuove righe, e che N indichi la prima riga libera di NATTAB. Descrivere il contenuto o la funzionalità delle altre variabili e funzioni/procedure eventualmente utilizzate.

SOLUZIONE:

```
while true
{  receive(II,DG);
   j=lookup(NATTAB, DG.indmitt,DG.options.sourceport,DG.inddest,DG.options.destport);
   if j ==-1     //DG non è relativo ad una sessione già instaurata
   {                                                    // va inserita una nuova riga
         NATTAB[N].prottrasp=DG.protocollo;                    //inserisci in NATTAB una riga nuova
         NATTAB[N].indirizzoesterno=DG.inddest;
         NATTAB[N].portaesterna=DG.options.destport;
         NATTAB[N].indirizzointerno=DG.indmitt;
         NATTAB[N].portainterna=DG.options.sourceport;
         if portalibera(DG.options.sourceport)
        {    NATTAB[N].portaNAT=DG.options.sourceport; }  //portaNAT è la porta usata dal router per la sessione
        else {    NATTAB[N].portaNAT=trovaportalibera();  }
        portalibera(NATTAB[N].portaNAT)=false;
         N++;
        }
DG.options.sourceport=NATTAB[j].portaNAT;
changeport(DG,NATTAB[j].portaNAT);
DG.indmitt=102.103.104.105;
   send(IE,DG);
 }
```